07

Extends

Transcripción

[00:00] No les voy a dar ningún spoiler del problema que vamos a tener. Vamos a seguir con la implementación de esto. Habíamos quedado que tenemos nuestra clase Contador que también es un funcionario, y si en este caso yo voy a crear un nuevo contador, vamos a ir aquí y vamos a crear aquí a nuestro contador. Contador, vamos a llamarlo Alexis. Contador Alexis igual new Contador. Perfecto.

[00:33] Y Alexis tiene un salario, setSalario de 5000. Y mismo caso que con Jimena, voy a necesitar crear un nuevo método aquí, no, aquí, para registrar el salario, ¿ahora de quién? Si ya registré del funcionario del gerente, ahora del contador. ¿Cierto? Entonces, "Ctrl + C", "Ctrl + V" y aquí en vez de gerente vamos a ponerle contador. Aquí, contador. Copiamos aquí, pegamos aquí para que compile y listo.

[01:16] Vemos que él compila correctamente. En este caso, lo que nosotros estamos haciendo aquí, registrar salario, tiene el mismo nombre en los tres casos pero el argumento es de tipo diferente. Este es un funcionario, este es para un gerente y este es para un contador. Esto se le conoce como sobrecarga del método. Cuando tú tienes un mismo método con un mismo método pero diferentes argumentos se llama sobrecarga.

[01:48] Es un concepto también nuevo pero que les va a ser muy útil. Volviendo al tema, ahora vamos a registrar el salario del contador en nuestro control de bonificación. Entonces le damos ControlBonificacion.registrarSalario, el salario de Alexis. Perfecto. Entonces ya tenemos aquí los tres salarios.

Guardamos todo, ejecutamos y vemos pues que el primer salario fue este, el segundo salario fue este de aquí y el tercer salario es este de aquí.

[02:20] Es decir, por cada vez que estamos registrando, nosotros estamos imprimiendo cuánto va de bonificación, entonces estamos obteniendo el resultado final aquí al último. Ahora, ¿cuál era la alerta que yo les estaba incentivando a ver? Estamos con la misma lógica del negocio en cada uno de estos métodos.

[02:47] Entonces digamos, si ahora yo voy a tener diseñador, ¿será que preciso otra vez crear un método de registrar salario para diseñador? ¿Será que si ahora yo tengo un analista necesito nuevamente crear un registrar salario para el analista específicamente? Si se están dando cuenta, tenemos otra vez el mismo problema que tuvimos al inicio antes de ver el concepto de herencia.

[03:13] Y es aquí donde polimorfismo tiene una utilidad muy importante, y van a ver por qué. Si yo agarro este método de aquí, registrar salario del gerente, y el de aquí abajo, registrar salario del contador y lo borro, me quedo solamente con registrar salario del funcionario. Perfecto. Le doy guardar. Si yo hago esto, vemos que mi clase TestControl sigue compilando correctamente. No me da ni un solo error.

[03:55] Aún cuando yo he eliminado los métodos para el gerente y para el contador. ¿Por qué? Porque el parámetro que le estamos mandando aquí es funcionario, que es digamos el tipo más genérico y le estamos mandando clases que son hijas de funcionario. Perfecto. Porque gerente es un funcionario, porque contador es un funcionario. Y bueno, funcionario es un funcionario.

[04:31] Ese es un claro ejemplo de aplicación de polimorfismo. Nos permite crear una única puerta de entrada por así decirlo, para todos aquellos elementos que sean funcionario. Porque por ejemplo, a ver, vamos a llevar esto un poco al mundo real. Tú no querías una puerta de entrada para un gerente y

otra puerta de entrada para un contador y otra puerta de entrada para un funcionario, ¿cierto?

[04:59] Todos entran por la misma puerta. ¿Por qué? Porque todos son funcionarios. Tú dices: "Por esta puerta están autorizados a entrar todos los que son funcionarios, todos los que cumplan con esa condición entran por aquí, sin importar quién sean". Entonces esto es lo mismo que estamos diciendo en este concepto.

[05:22] Todos los que sean funcionarios van a ser aceptables, elegibles, para entrar como parámetro en este método. ¿Por qué? Porque es de la clase más genérica, por lo tanto si yo vuelvo a ejecutar acá mi test de control de bonificación, voy a ejecutar aquí y tenemos el mismo resultado final. Igual está en 100, 10600 y 10850.

[05:46] Vemos que no ha variado en nada, el cálculo está bien hecho. ¿Por qué? Porque simplemente mi método está aceptando a la clase más genérica, por lo tanto todos los que hereden de esta clase son elegibles para ejecutar este método.